CHAPTER 7

NODE-CENTRIC HYBRID ROUTING FOR WIRELESS INTERNETWORKING

J.J. Garcia-Luna-Aceves and Soumya Roy
Computer Engineering Department
University of California, Santa Cruz *
jj@cse.ucsc.edu, soumya@cse.ucsc.edu

7.1. Introduction

Multihop packet radio networks (or ad-hoc networks) consist of wireless routers that interconnect attached hosts without the need of any pre-established communication infrastructure. These networks play an important role in relief scenarios, battlefields and conference scenarios, where there is no base infrastructure.

Table-driven or proactive routing protocols can incur excessive signaling overhead in large ad-hoc networks (e.g., networks with hundreds of nodes or more), because each node in the network must maintain routing information for every other network node, even if the node never needs to handle traffic destined for some nodes and the path between any two nodes in a highly mobile environment changes frequently. Control overhead in proactive routing protocols increases with the size of the network and becomes redundant if the number of communicating peers is much less than the total number of nodes in the network.

To address the scaling problem of table-driven routing, on-demand routing protocols have been proposed for ad hoc networks. Nodes running such protocols set up and maintain routes to destinations only if they are active recipients of data packets and generate network-wide queries to establish routes to destinations. However, when only a few nodes of the ad hoc network must act as sources and sinks of data packets, and such choices are very stable, maintaining routing information to such destinations on demand and treating those nodes as any other

Report Documentation Page		*Form Approved* *OMB No. 0704-0188*
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.		

1. REPORT DATE **2006**	2. REPORT TYPE	3. DATES COVERED **00-00-2006 to 00-00-2006**
4. TITLE AND SUBTITLE **Chapter 7. Node-Centric Hybrid Routing for Wireless Internetworking**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**26**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

node may not be as attractive as a proactive approach. This motivates the interest in a hybrid approach to routing in ad hoc networks.

The Zone Routing Protocol (ZRP) [Haas and Pearlman, 1999] constitutes a framework for hybrid routing in ad hoc networks. ZRP adapts a hierarchical-routing approach based on clusters (called zones) and maintains routes proactively to destinations inside a zone, and on-demand routing is used to establish routing information spanning more than one zone. In this chapter, we advocate a different approach to hybrid routing that is node centric rather than based on zones or areas of the network.

The rationale for a node-centric approach to hybrid routing is that there are many cases in which certain nodes in an ad hoc network need to host special services to other nodes in the ad hoc network, specially when the ad hoc network is a wireless extension of the Internet (e.g., DNS services, Internet access, web proxies). We call those nodes that support special services for the rest of the nodes (and therefore that have a high likelihood of communicating with the rest of the ad hoc network) *netmarks*. This scenario is illustrated using Fig. 7.1, which shows an ad hoc network of mobile nodes $a, b, \ldots\ldots$, and t and a single netmark. The netmark can be fixed as well as mobile, depending on the application scenario. Under a node-centric hybrid routing approach, paths are constantly maintained between nodes $a, b, c, \ldots$, and t and the netmark. The forward and reverse paths between netmarks and nodes $a, b, c, \ldots$, and t (shown using solid lines) are maintained constantly. If node e needs to communicate with node f, while node j communicates with node p, the paths between those nodes are set up on demand and are shown in dashed lines in Fig. 7.1. Observe that node c does not need to know how to reach node r, so no route needs to be maintained at node c to reach node r.

The landmark hierarchy [Tsuchiya, 1988] is an earlier node-centric approach to hierarchical routing designed for proactive routing in large networks. The key difference between the node-centric routing described in this paper and the landmark hierarchy is that a landmark becomes the address of a common node, while a netmark is a destination that provides services. Hence, the node-centric routing is an orthogonal approach to the landmark hierarchy, which is aimed at aggregating routing-table entries.

Section 7.2 introduces two approaches to node-centric hybrid routing, in which a netmark is distinguished from normal nodes (which can be done through addressing or by having an explicit tagging mechanism). In one approach, a netmark forces the rest of the nodes to maintain their routes to it for long periods of time once they acquire such routes. This amounts to extending the caching of netmark routing information. In

another approach, a netmark uses proactive routing updates to push its routing entry into the routing tables of the rest of the nodes in the ad hoc network. To describe our approaches, we assume that the nodes in the ad hoc network form a subnet and each node has a unique identifier, by which routing protocols and other applications can identify it. By looking at the address of the destination of any IP packet, any node can determine whether or not the packet is meant for a node outside the subnet. Links are assumed to be bidirectional, such that when a node hears from another node, it can assume that it can also forward packets to that neighbor. Nodes are assumed to operate correctly and information is assumed to be stored without errors.
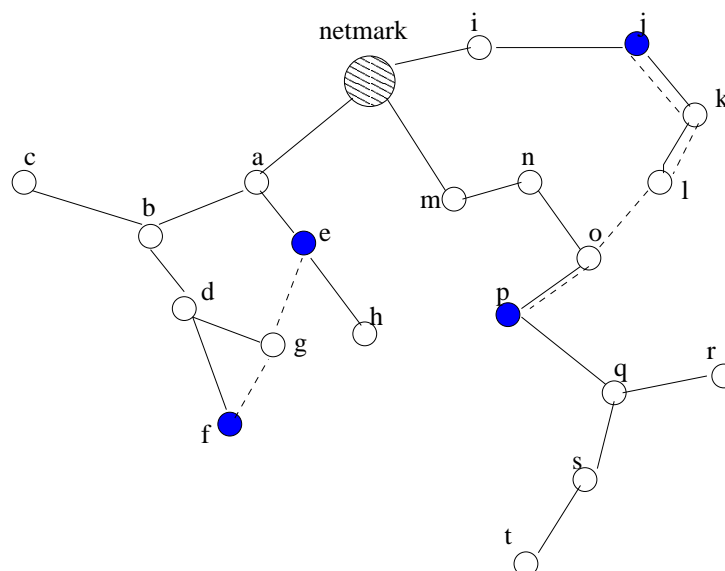

Figure 7.1. Figure showing an ad hoc network with a single netmark

Section 7.3 shows how an on-demand routing protocol can be modified to adopt the node-centric approaches proposed in Section 7.2. Our choice of using on-demand routing based on partial link-state information for node-centric hybrid routing is based on the performance of on-demand link-state routing protocols and the ease with which node centric hybrid routing mechanisms can be incorporated in protocols based on link-state information. Section 7.4 addresses the issues that arise with having multiple netmarks in an ad hoc network. Section 7.5 presents the results of our performance comparison of node-centric hybrid routing with purely on-demand routing protocols. For our study, we use the ad hoc on-demand distance vector routing (AODV) protocol [Perkins

et al., 2002], the dynamic source routing (DSR) protocol[Johnson and Maltz, 1994] and the source tree on-demand adaptive routing (SOAR) [Roy and Garcia-Luna-Aceves, 2001] protocol. Previous simulation studies [Roy and Garcia-Luna-Aceves, 2001], [Broch et al., 1998], [Das et al., 2000] have reported the performance of on demand routing protocols in mobile ad hoc networks with a very uniform traffic pattern, i.e., traffic flows are randomly distributed throughout the network, with no node being accessed more than others. However in practice, traffic patterns are not uniformly distributed and typically concentrate around nodes that offer special services to others, even though all nodes combine to forward data packets. This is true specifically when an ad hoc network is a wireless extension of the Internet and a few network access points are used to attach to the Internet as well as to provide such services as DNS and proxies. The traffic (mostly web traffic, which can be assumed to be heavy-tailed ON-OFF traffic) would exist mainly between the nodes of the ad hoc network and the network access points. This is also the case of battlefields or relief scenarios, in which a large amount of communication exists between a group leader and the rest of the group. The results of our simulation experiments illustrate the benefits of the node-centric hybrid routing approach.

7.2.   Node Centric Hybrid Routing

7.2.1   Hybrid Routing by Extended Caching of Netmarks

In pure on-demand routing protocols, routers set up paths to other nodes based on the existence of flows with them. Routes are cached once they are obtained using a route discovery mechanism and they are modified when they become invalid due to link failures. Among the on-demand routing protocols proposed in the literature, the basic difference is in how routes are cached and invalidated, and how route changes are reported to other nodes. Extending an on-demand routing protocol to support hybrid routing through extended caching of the routes to the netmarks entails the following two main changes to a pure on-demand protocol:

- A node sends a route request for a netmark whenever it looses all routes to it independently of its traffic to the netmark.

- Any node generates a route error whenever it detects the loss of a route to a netmark independently of the traffic for the netmarks.

We now summarize how extended caching can be applied to specific on-demand routing protocols, namely AODV, DSR and SOAR.

The ad-hoc on demand distance vector (AODV) protocol is based on distance vectors and uses sequence numbers to prevent temporary and permanent loops. AODV supports incremental routing because no routing table loops can be formed. Route Requests (RREQs) are generated by the sources of data packets and forwarded by intermediate nodes. When RREQs are forwarded, reverse routes for the source of the RREQ are installed. Route replies (RREPs) can be sent by the destination or an intermediate node with an unexpired route entry for the destination. The RREP message initiates the creation of a path for the destination in intermediate nodes that forward the RREP back to the sender of the RREQ. Each routing table entry has an expiration period (*active_route_timeout*) associated with it.

The specification of AODV [Perkins et al., 2002] states that AODV can use the periodic network-layer Hellos or link-layer notifications for determining connectivity with neighbors. When a node detects that its path to a destination has been broken due to a failed link, it sends a route error (RERR) packet to its active predecessors for that destination, i.e., those neighbors known to use the node to forward packets to the destination.

AODV can easily incorporate the idea of extended caching by increasing the expiration period of the route for the netmarks to a very high value or infinity. Every time the paths to a netmark fail due to link failures, the nodes can send RERRs to active predecessors, which travel upstream. When a node looses its route for a netmark, it starts the route discovery mechanism, irrespective of the presence of traffic for netmarks. Royer et al. [Royer et al., 2001] have suggested a similar foreign agent discovery mechanism using AODV where routes for foreign agents are discovered irrespective of the presence of traffic. However, routes for foreign agents are not maintained differently from the routes to common nodes.

The dynamic source routing (DSR) protocol uses source routes to forward data packets and exchanges routes in the form of paths. Routes are stored in a cache, until an indication that a link in the route is broken is obtained through route error (RERR) messages or link-layer notifications. A route discovery cycle is started by a source if it looses all routes to the required destination. DSR can determine on its own if a link is broken by doing multiple retransmissions, or can depend on the link layer for link-failure notifications during packet transmission failures. Route error containing information about failed links are sent towards the source of data packets. In DSR, if a node detects the route failure for a netmark while forwarding data packets for a common node, then it does not know how to propagate RERR information for the netmark.

Hence, generating route errors independently of traffic to netmarks is not trivial in DSR and flooding RERRs would be expensive. RREQs sent in response to route failures to netmarks cannot be used to replace the functionality of RERRs, because RREQs do not have the information about failed routes. However as in AODV, anytime the route to a netmark is broken, the basic route discovery mechanism of DSR can be easily modified to send RREQs for netmarks, irrespective of the presence of traffic for them.

The source-tree on demand adaptive routing (SOAR) [Roy and Garcia-Luna-Aceves, 2001] protocol is a link-state routing protocol in which routers exchange minimal source trees in their control packets. A minimal source tree consists of the state of the links along the paths used by the routers to reach active (important) destinations. Important destinations are active receivers, relays or possible relays. Each router uses the minimal source tree of its neighbors and its outgoing links to get the partial topology of the network. Routing table entries for known destinations are computed using a path selection algorithm on the partial topology and the data packets are forwarded hop-by-hop according to the routing entries. Links are validated using sequence numbers. SOAR uses queries and replies to create routes to unknown destinations. Update packets containing modified minimal source trees are generated when a node decides that its neighbors need to be updated to prevent erroneous forwarding or the formation of routing-table loops in the paths to important destinations.

The notion of *important node* in SOAR can be generalized to incorporate the concept of netmarks by always tagging netmarks as *important*, while the rest of the nodes become important on the basis of the traffic flowing to those nodes. Another variation of this approach is that netmarks would be considered important for longer periods of time than common nodes. Therefore, depending on the level of importance of a particular node, paths to nodes remain fresh for different time-spans. Hence, a simple modification to the basic mechanism of routing information exchange in SOAR enables incorporating the idea of extended caching of routes to netmarks. We call this modification netmark-aware on-demand link state routing (NOLR).

7.2.2   Hybrid Routing with Proactive Routes to Netmarks

The second approach to hybrid routing consists of maintaining proactive routes for the netmarks, while on-demand routes are used for other

nodes. The modifications required for any on-demand routing protocol to adopt this approach are the following:

1 Adding a route for a netmark for the first time necessitates sending updates to neighbors, so that they can also set up new paths to the netmarks.

2 Depending on the protocol, route errors, route requests, or route updates are generated for netmarks, independently of the traffic to them.

3 A netmark can advertise its presence by sending Hellos to enable new neighbors to set up paths to it, and to find out whether the netmark is still reachable without having to depend on link-layer notifications. This enables paths to netmarks to be more proactive, rather than being data-packet driven.

Next we summarize how DSR, AODV or SOAR can be changed to incorporate the above concepts of hybrid routing. Implementing a network-layer Hello mechanism at the netmarks is easy in any of the three protocols.

As mentioned earlier, AODV uses destination sequence numbers to validate routes to destinations. Hellos sent by the netmarks in AODV must contain the highest sequence number for the netmarks, so that the receiving node can install new direct routes for the netmarks. Given that DSR sends RERRs only to the source of data packets when there is a failure of packet transmissions along a particular link, adding a Hello mechanism will not help DSR, because in such a case the protocol does not have a mechanism to decide how to propagate information about the loss of netmark-routes to other nodes in the network. However, if a node keeps track of the neighbors (i.e., predecessors) that use it for data delivery to netmarks for the last *pre-defined* amount of time, route failures to netmarks can be recursively reported to other nodes through the predecessors. Incorporating the Hello mechanism benefits both AODV and SOAR, because the paths to netmarks remain more up to date.

To propagate new route information for netmarks in SOAR requires only a change in the rules for sending an *update*. Rather than only sending an *update* for a destination when the path cost to it increases, updates are also sent when a route for a new netmark is discovered. Unlike SOAR, both AODV and DSR need the introduction of a new type of control packet that propagates route updates for netmarks to all the nodes in the network when a route to a netmark is first discovered.

7.3.   Hybrid Routing Using SOAR

We have chosen SOAR as the basic routing protocol to illustrate the benefits of node-centric hybrid routing over on-demand routing because of the following reasons because SOAR has been shown to be more efficient than DSR [Roy and Garcia-Luna-Aceves, 2001] and the results presented in Sec. 7.5 show that SOAR outperforms AODV. Furthermore, the modifications needed in SOAR to adopt hybrid routing are much simpler than the modifications required in DSR and AODV.
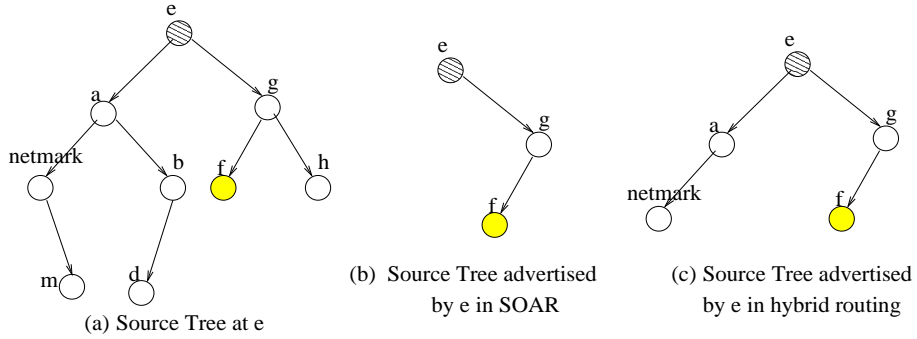

(a) Source Tree at e

(b) Source Tree advertised by e in SOAR

(c) Source Tree advertised by e in hybrid routing

Figure 7.2.   Figure showing difference in control information in SOAR and NEST/NOLR

Extended caching of netmarks can be adopted in SOAR by considering paths to different destinations as important for different periods of time. This modification of SOAR is called NOLR (netmark-aware on-demand link state routing).

The Netmark Enhanced Source Tree (NEST) routing protocol adopts the routing mechanisms of SOAR to maintain proactive routes for netmarks and on-demand routes for other nodes in the network. We provide the details of NEST in the rest of this section.

NEST and SOAR both exchange link-state information in the form of the minimal source trees, that contain the state of the links to reach important destinations in the network. However, there is a slight difference in the actual contents. In NEST, the advertised minimal source tree always contains a path to the netmark. Fig. 7.2 illustrates the difference in the control message advertised by node e in the network shown in Fig. 7.1, in which every node has a proactive path with the netmark and nodes e and i have on-demand routes set up for f and p, respectively. The source tree at node e (Fig. 7.2(a)) is the tree consisting of links that node e uses to reach the netmark and other nodes in the network. Node e advertises a portion of this complete source tree to its neighbors, which is called the *minimal* source tree. For SOAR, the *minimal*

source tree would only consist of links needed to reach nodes with which it has active flows. In this example, node e has active flow with node f, the minimal source tree advertised by node e would be as shown in Fig. 7.2(b). In NEST, even if node e does not have active communication with the netmark, it advertises links in the path to it (as shown in Fig. 7.2(c)).

7.3.1   Netmark Discovery

In NEST, netmarks send Hello packets to inform their neighbors of their presence. This same effect is achieved by sending beacons at the MAC layer. At the routing layer, if the node does not receive the Hello packet for some predefined interval of time, then it can declare that the link to its neighbor is down. The link layer can also notify the routing layer about link failures when it cannot deliver data packets. In the presence of data traffic, this later mechanism can detect link failures faster.

When a node has a new entry for the netmark, it updates it neighbors about the new route using an update. Therefore, every node in the network eventually knows about at least one path to the netmark. For example, in Fig. 7.1, when a learns about the existence of the netmark in its neighborhood, it advertises its path to it, and its neighbors b and e re-advertise. Updates in NEST are broadcast packets and hence unreliable. If updates are lost, some nodes may not know about the route to the netmark. Furthermore, when a node comes up, it is not aware of netmarks. Under such circumstances, a node must initiate a *query*. However, because paths to netmarks are maintained proactively by all common nodes, there is a considerable reduction of queries required to establish routes to netmarks.

Another important issue regarding tagging some nodes as *netmarks* is how to disseminate the information about the nodes with netmark status. This can be achieved in one of the following ways:

- All common nodes can be pre-configured statically with the host addresses of the netmarks. This approach is beneficial if some nodes maintain statically their netmark status and all common nodes know about the netmarks. For example, netmarks hosting proxy services or DNS services will not change and the mobile routers can be statically configured with the address of the netmark.

- If the assignment of netmark status to nodes is not fixed, then the new netmarks can advertise their status by sending Hello Packets and all the nodes can mark the netmarks in the minimal source

trees while sending updates. Marking the netmarks means a special flag is attached in the minimal source trees to links with netmarks as tails. When a node ceases to be a netmark, it can notify all nodes that it is no longer a netmark by flooding. This situation can happen in relief scenarios or battlefields, where the group leaderships may change over time.

7.3.2   Maintaining Paths in NEST

Figure 7.3.   Setting up of paths between netmarks and other nodes

Fig. 7.3 illustrates how the forward and reverse paths between a common node and netmark are set up. In Fig. 7.3, when node c learns of the netmark, it advertises the netmark in its source tree and hence node b knows about the netmark and neighbor c. When node b advertises its own tree, it reports the information about nodes b, c, and the *netmark*, i.e., the path to netmark consisting of the intermediate nodes b, c. Similarly, node b learns about an alternate path $[b, d, e, netmark]$ from node d, but node b chooses the path through node c because it has a smaller length. If link (b, c) fails, then node b can choose the alternate path to the netmark through node d. The downstream nodes from a mobile node towards the netmark may only know about the upstream predecessor and not know about all upstream nodes (e.g., when node b advertises its source tree, it may not advertise link to node a because node a is not

an important destination, in such a case, node c knows only about node b, but not about node a). Similarly, the netmark may know about node c, but not about nodes a and b. In order to send packets to node a, the netmark would have to send a *query* for node a in such a case. To prevent a *query* to be initiated from the netmark for every mobile node in the network, the following mechanism is adopted to set up the reverse paths from a netmark to any common node without introducing any extra control overhead.

When data packets start flowing from a node towards the netmark, the intermediate nodes along the path towards the netmark can set up paths towards the source of the data packets. For example, when the data packet from node a reaches node c and finds the destination is a netmark, then node c adds an entry in its routing table for node a as $[dest = a, nexthop = b]$. Similarly, the netmark keeps a routing-table entry $[dest = a, nexthop = c]$. These routing entries expire after a *Soft_State_Interval*. When link (b, c) breaks, data packets are forwarded along the path $[a, b, d, e, netmark]$ and the netmark replaces the entry $[a, c]$ with $[a, e]$ when the data packets arrive from node e. Similarly, when nodes d and e forward packets, they set up soft-state entries for destination a. Node c removes entry $[a, b]$ after the *Soft_State_Interval* due to the absence of any data packets from node a towards the netmark. Routes for different netmarks from the same common node can pass through same intermediate nodes. In that case, at any intermediate node, variations of the path traced by the data packets with the same source but with different destinations can lead to route flapping. Therefore, to prevent route flapping, the soft-state entry is not modified for $Soft\_State\_Interval$ after the last change.

The reverse routes are set up towards the source based on the flow of data packets only if the destination of the data packets is a netmark. This is because each node maintains up-to-date paths only to netmarks. Hence, given that the paths from a node to any other node may not be current, the reverse path between any two common nodes would lead to data packet losses. In steady state, the reverse route from a netmark is essentially the same as the forward path. Therefore, if the forward routes to the netmarks are correct, the reverse routes are going to be also correct.

As discussed above, the reverse routes are set up using the path traversed by the data packets, without incurring extra control overhead. However, in the absence of a flow of data packets from a common node to the netmark, the netmark must resort to queries to find paths to the destinations. The number of these queries is reduced drastically if the flow between the netmark and the node is mainly bi-directional. This

mechanism of reverse path set up and maintenance ensures symmetry of the paths taken by data packets in *tcp*-like bi-directional flows.

The path maintenance mechanism is similar in both NEST and SOAR; however, paths for netmarks in NEST are always maintained up-to-date, because the netmarks are always tagged as important. When a router finds that the distance to an *important* node increases, it reports modified minimal source trees in its updates to its neighbors in order to correct their link-state information. Link-state information is validated using sequence numbers. A link-state update is trusted if it has a higher sequence number, or the same sequence number but a smaller link cost.

When a node receives a packet from the application layer and it is meant for a node in the ad hoc network, it forwards it to the next hop as indicated in the routing table entry, provided that the node has a route to the destination. All nodes can determine when the packets are meant for a node outside the ad hoc network by looking at the IP address of the destination, and forward such packets towards the netmark.

7.4.   Multiple Netmark Scenarios

When multiple netmarks are present in the network, the way in which nodes affiliate themselves to netmarks has a direct impact on performance of routing protocols.

A node can be made to always communicate to a particular netmark irrespective of the location of the node with respect to the netmark in the network. This static affiliation can happen in a battlefield scenario or sensor net, where irrespective of the position of a soldier or sensor, data must be reported to a specific group leader or collection station. Though every node is only affiliated to a particular netmark, it has to know how to reach every netmark in the network, because every node may have to forward data packets for other nodes in the network and all nodes may not be affiliated to one particular netmark..

In a different scenario a node need not be affiliated with any particular netmark and it can be allowed to communicate with any netmark. This dynamic affiliation can happen when the ad-hoc network is an extension of the Internet, and there are multiple Internet access points. The common nodes in the network can communicate with any access point, because all access points are connected to the Internet. Given that packets are forwarded to any netmark, routing becomes efficient in terms of control overhead because (a) redundancy of routes to the Internet reduces the number of expensive route discovery cycles; (b) outgoing packets within the ad hoc network tend to be more optimal; and (c) an anycast route discovery mechanism can help to reduce control

overhead, improve optimality of paths and reduce latency of path discovery. Queries are not required to be sent individually for each netmark. Instead *anycast* queries can be sent asking for a route to the anycast address of all the netmarks. In such a case, any router with a route to any one netmark can reply. The reply would contain the route to the nearest netmark if more than one routes were known to the responding router.

There can be scenarios in which netmark affiliations are both static and dynamic. Packets for some fixed destinations outside the ad hoc network are always forwarded to some particular netmarks, while packets for others can be forwarded to the nearest netmark.

Depending on how data packets enter the ad hoc network and how the outgoing packets are forwarded, paths followed by data packets can be symmetric or asymmetric. In the case of static affiliation, as shown in Fig. 7.4(a), if the incoming packets for a mobile router are always forwarded to the netmark with which the router is statically affiliated, then only the data-path can be symmetric. For the example shown in Fig. 7.4(b), due to the dynamic affiliation, though data packets arrive for router i from *netmark 1*, the outgoing packets are forwarded towards *netmark 2*, thereby making the data-path assymetric.

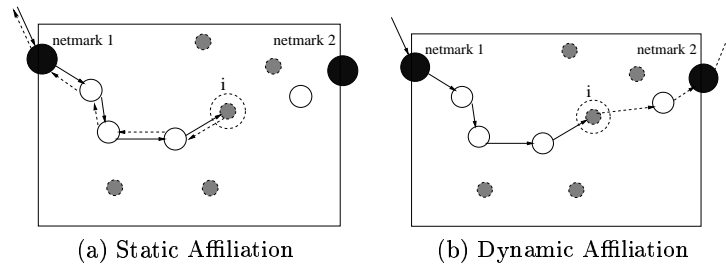(a) Static Affiliation   (b) Dynamic Affiliation

Figure 7.4.   Figure depicting paths taken by data packets in an ad hoc network with multiple netmarks

When the netmark is an Internet access point, it advertises routes to subnets but not host routes to the Internet. Accordingly, in the example of Fig. 7.4 it may happen that *netmark* 1 advertises path to a subnet that includes node i. However, due to network partitioning, *netmark* 1 may not have any route for node i. Hence, packets for node i are dropped if they are forwarded to *netmark* 1. However, node i can still be reached through *netmark* 2. This can be remedied if the netmarks form a fully-connected overlay network using the wired Internet. In such a case, packets for node i can be forwarded using the overlay network from *netmark* 1 to *netmark* 2 and *netmark* 2 can finally deliver the data packets to node i.

7.5.   Performance Evaluation

7.5.1   Simulation Model

We have compared the performance of node-centric routing approaches NEST and NOLR with the performance of pure on-demand routing protocols SOAR [Roy and Garcia-Luna-Aceves, 2001], DSR [Johnson and Maltz, 1994] and AODV [Perkins et al., 2002] using the ns2 network simulator. For DSR, we used the code available with the ns2 simulator [ISI-NS, 2000]. For AODV, we used the code available from the implementation of Marina[Marina, 2000] and the constants provided with the code. SOAR has been implemented according to the specifications provided by Roy and Garcia-Luna-Aceves [Roy and Garcia-Luna-Aceves, 2001]. NOLR is the modification of SOAR with extended caching of routing information for netmarks. NEST uses the same constants used in SOAR [Roy and Garcia-Luna-Aceves, 2001] along with three additional constants: Hello_Interval, Dead_Time_Interval and Soft_State_Interval. Hello_Interval (three secs) is the interval between sending of two consecutive Hello packets by the netmark. Dead_Time_Interval (nine secs) is the time interval for which if a node does not receive any packet from a netmark, the link to the netmark is considered to be down. When a broadcast control packet is sent, the netmark can defer the next transmission of Hello packet for a time equal to the Hello_Interval. Soft_State_Interval (one sec) is the maximum time a soft-state routing entry stays in the routing table, without being refreshed.

DSR, AODV, SOAR, NOLR and NEST do not depend on the link layer for neighbor discovery. All protocols use link-layer indications about link-failures when data packets cannot be delivered along particular links. Use of link-layer information for discovering neighbors can significantly improve the performance of routing-layer protocols. However, because our objective is to test the routing protocols as stand-alone protocols, we have not considered the effects of MAC layer interactions on the routing protocols' performance and promiscuous mode of operation has been disabled. The link layer protocol used is the IEEE802.11 distributed co-ordination function (DCF) for wireless LANs, which uses a RTS/CTS/DATA/ACK pattern for all unicast packets and DATA packets for all broadcast packets. The physical layer approximates a 2 Mbps DSSS radio interface. The radio range of the radio is 250m. We assume a netmark does not change its *netmark* status during the entire length of the simulation.

Nodal movement occurs according to the random waypoint model introduced in [Broch et al., 1998]. In this model, each node is at a random point at the start of the simulation and after a *pause time* seconds the

node selects a random destination and moves to that destination at a constant speed. Upon reaching the destination, the node pauses again for *pause time* seconds, chooses another destination, and proceeds there. The speed of a mobile node during its movement is uniformly distributed between 0 and 20m/sec.

We have introduced two traffic models for performance evaluation, which we call (a) the INTNET model and (b) the RELIEF model. These traffic models are more realistic compared to the traffic models used in prior analyses [Broch et al., 1998], [Das et al., 2000], in which continuous CBR traffic flows exist between randomly chosen nodes making the traffic pattern more or less uniform throughout the network.
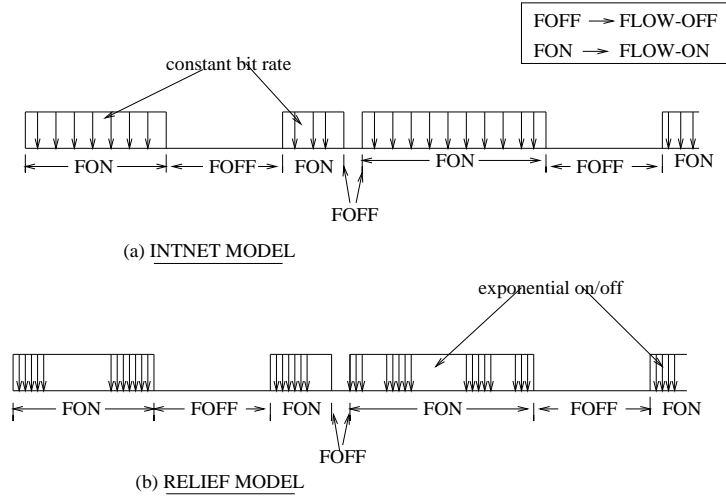

Figure 7.5. Traffic Flow Models

The INTNET model is similar to the scenario of using ad hoc networks as wireless extensions of the Internet. The communication is mainly from each of the common nodes towards the netmark hosting commonly-accessed servers or acting as the access point to the Internet. The number of flows between mobile nodes only is much less compared to the number of flows between nodes and netmark. The traffic pattern is based on a FLOW_OFF/ON model, as shown in Fig. 7.5(a), with the parameters as given in Table 7.1.

During the FLOW_ON period, there exists *cbr* traffic and there is no packet flow during the FLOW_OFF period. The motivation behind simulating the FLOW_OFF/ON model, rather than a model in which the flows are on continuously, is that Web traffic consists of FLOW_ON/OFF periods, where the OFF periods correspond to the user's think time, and

FLOW_ON period	:	Uniform Dist (30,120) secs
FLOW_OFF period	:	Uniform Dist (50, 120) secs
Packet Size	:	66 bytes
Rate	:	3, 5 packets/sec per node

Table 7.1.   Constants for Flows in INTNET model

the ON period represents download time. In our experiments with the INTNET model, there are four random flows between any two randomly selected common nodes at any time. The duration of these flows is always 200 secs and all the flows are bi-directional in nature.

The RELIEF traffic model is used to simulate traffic in relief or battlefield scenarios, where the group members report to the group leaders while the group members also exchange information. The group leader is the netmark contacted more frequently compared to other nodes. There are four random flows between common nodes and there are at most six random flows from a common node towards the netmark. We divided the set of common nodes into five groups and only one member in the group can talk at a time with the netmark. The traffic pattern per group is also like the FLOW_OFF/ON model. The packet arrivals during FLOW_ON period follow an interrupted deterministic process (IDP) as shown in Fig. 7.5. The IDP model is used to simulate the voice traffic. The ON/OFF periods during a FLOW_ON period correspond to talkspurt/silence periods of the speaker. The parameters for the RELIEF model are as given in Table 7.2.

FLOW_ON period	:	uniform (30,150) secs
FLOW_OFF period	:	uniform (10, 20)
Packet Size	:	66 bytes
Rate	:	17 packets/sec (9kbps)
talkspurt	:	350ms
silence	:	650ms

Table 7.2.   Constants for Flows in RELIEF model

We evaluate the routing protocols based on packet delivery ratio, control packet overhead, average hop count, end-to-end delay, and the number of queries and replies sent by each protocol.

7.5.2   Experimental Scenario 1

The first scenario consists of a network of 31 nodes moving over a rectangular area of 1000mx500m. There is a single netmark in the system, which is placed at coordinates (500, 250) and is fixed throughout the
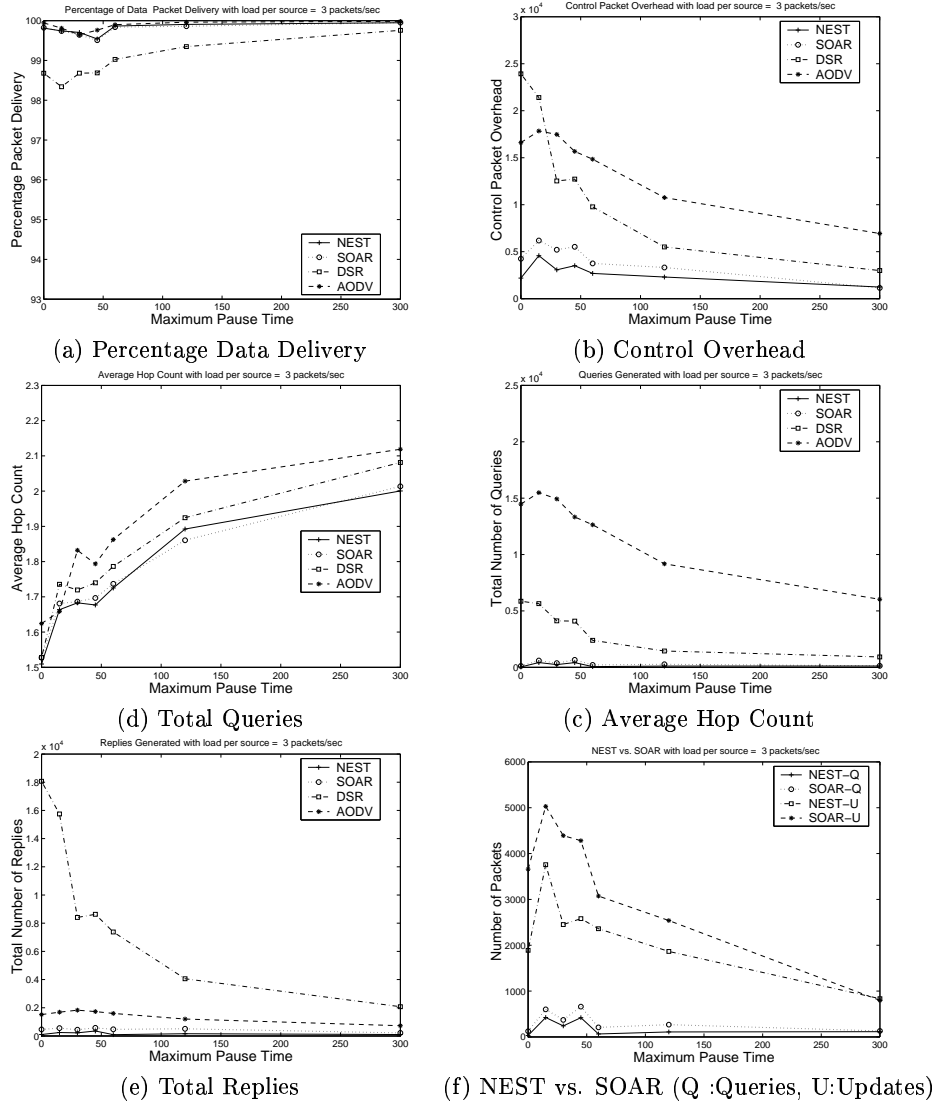
(a) Percentage Data Delivery

(b) Control Overhead

(d) Total Queries

(c) Average Hop Count

(e) Total Replies

(f) NEST vs. SOAR (Q :Queries, U:Updates)

Figure 7.6. Performance of NEST, SOAR, DSR, AODV in a 31node Network at load per node of 3 packets/sec with fixed netmark

simulation time. The pause time of other nodes is uniformly distributed between zero and a maximum value, which can be one of 0, 15, 30, 45, 60, 120 and 300 seconds. The simulation length is 600 secs, while the results are presented on the basis of at least 3 simulation runs, where each run is with the same INTNET traffic model but with a different randomly generated mobility scenario (this is also true for subsequent
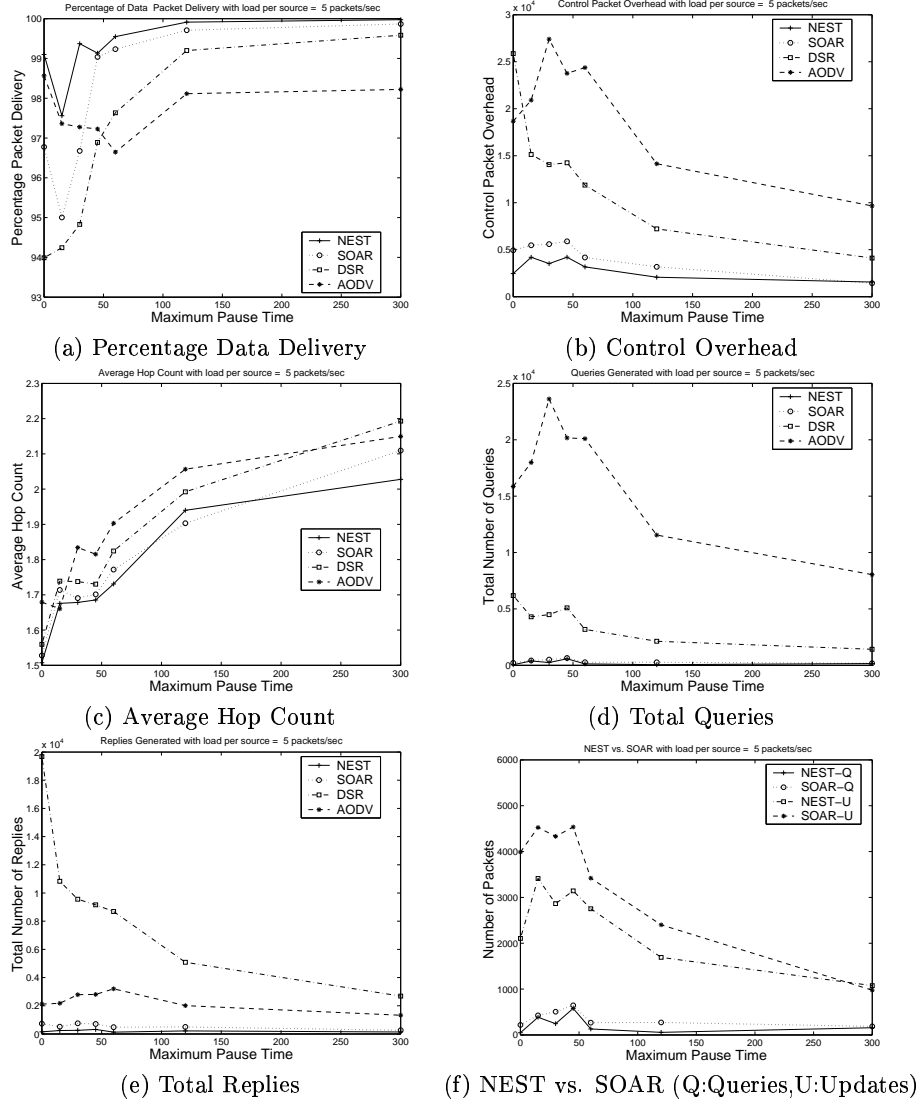
(a) Percentage Data Delivery

(b) Control Overhead

(c) Average Hop Count

(d) Total Queries

(e) Total Replies

(f) NEST vs. SOAR (Q:Queries,U:Updates)

Figure 7.7.   Performance of NEST, SOAR, DSR, AODV in a 31node Network at load per node of 5 packets/sec with fixed netmark

experiments). Performance results are presented for two different load scenarios of three and five packets/sec per node during the FLOW_ON period.

Most of the results for AODV, SOAR and DSR conform to the results published previously for those protocols [Das et al., 2000], [Roy and Garcia-Luna-Aceves, 2001] and [Broch et al., 1998]. As shown in

Fig. 7.6(b) and Fig. 7.7(b), AODV's control overhead is found to be significantly higher than DSR's or SOAR's, except for the high mobility scenarios. AODV's control overhead consists primarily of queries (Fig. 7.6(d) and Fig. 7.7(d)), while the control overhead of DSR consists mainly of replies(Fig. 7.6(e) and Fig. 7.7(e)). This is because AODV resorts to the route discovery mechanism more often than DSR, while DSR sends multiple replies to queries. Contrary to the findings in [Das et al., 2000], [Broch et al., 1998] an interesting result for the INTNET model is that AODV's control overhead in highly mobile scenarios is lower than DSR's. Because each node in the INTNET model sends and forwards packets for a netmark, the number of cached entries for the netmark is comparatively higher in DSR than in scenarios where the traffic pattern is uniform. That effectively leads to significantly higher number of cached replies (many of which contain stale routes) which amount to higher control overhead in DSR than in AODV. In low mobility scenarios in which path information becomes stale less often, the effect of injecting old routes due to multiple replies is much smaller.

SOAR produces much fewer control packets compared to DSR or AODV under all mobility scenarios with varying loads. The reason behind this is that SOAR resorts to fewer route discovery queries than AODV or DSR, because of the redundancy in the exchanged routing information in the control packets specifying minimal source trees, and because of the use of mostly local updates to solve path breakage, rather than sending route error messages to the source of data packets. Because SOAR and DSR can use stale information, SOAR and DSR deliver slightly fewer data packets compared to AODV under heavy load scenarios and high mobility. The performance degradation is less in SOAR compared to DSR, because SOAR uses sequence numbers to validate link−state information and DSR uses explicit route error messages to invalidate link information.

The performance of NOLR was found to be almost identical to SOAR's. Accordingly, for clarity, Fig. 7.6 and Fig. 7.7 do not show the results obtained for NOLR. Unlike SOAR, NOLR maintains routing information for netmarks for longer periods of time compared to the time for maintaining information for other nodes. The reason why NOLR and SOAR exhibit the same performance in this scenario is that each node either sends or forwards packets for the netmark the vast majority of time; Therefore, any node in SOAR ends up treating the netmark as *important* throughout the simulation.

Under all scenarios, NEST performs much better compared to all purely on-demand routing protocols, both in terms of data delivery and control overhead. NEST (Fig. 7.6(a) and Fig. 7.7(a)) always delivers

more packets compared to other protocols, with the effect being more prominent under heavy load. Overloading the network does not affect NEST, because each node always maintains correct paths to the netmark. Therefore, under heavy loads, NEST looses much fewer data packets than AODV, even though AODV attempts to avoid using stale routing information. SOAR maintains information for netmarks for significant periods of time; however, NEST paths are more accurate, because the netmark advertises itself periodically to force its routing information in other nodes and nodes using NEST update their neighbors when they first discover routes to netmarks. This conclusion is validated by the results of Fig. 7.6(f) and Fig. 7.7(f), in which we see that more updates are needed in SOAR compared to NEST to purge stale link-state information. On an average, NEST produces around 30% fewer updates than SOAR. We also find that NEST (Fig. 7.6(f) and Fig. 7.7(f)) produces fewer queries compared to SOAR, which leads to a reduction of replies in NEST (Fig. 7.6(e) and Fig. 7.7(e)). Queries are still sent by NEST for discovering routes on-demand with common nodes and for probing the netmark when the netmark becomes unreachable due to network partitions. We also see from Fig. 7.6(c) and Fig. 7.7(c) that the average hop count in NEST is the smallest, because NEST detects the presence of netmarks much faster. However, to conserve network bandwidth, given that the routers in NEST do not advertise route changes when distances to any node decrease, the path length in NEST can still be sub-optimal.

7.5.3   Experimental Scenario 2

The second scenario focuses on the effect of netmark mobility. It consists of a network of 30 nodes and one netmark with common nodes moving at a speed uniformly distributed between 5m/s and 20m/s. Pause times are uniformly distributed between 0secs and 30secs. Three different movement scenarios for the netmark are analyzed while keeping the mobility pattern for other nodes the same. The netmark in these scenarios is either static (model s), mobile (model m, the netmark moves over a rectangular area (250,250)) or very mobile (model vm, the netmark can move over the entire area (1000,500)). Because most of the traffic is towards the netmark, the routing protocols would be more stressed to maintain routes as the mobility of the netmark increases. The netmark moves with a speed similar to the speed of common nodes with pause time between 10 and 30 secs. We use the INTNET and RELIEF traffic models, which are indicated as REL and INT in Fig. 7.8. Accordingly, a static netmark model with INTNET traffic pattern is indicated as

sINT, while a *vm* model with RELIEF traffic pattern is represented as *vmREL*, for example.

For this scenario, the results for AODV are significantly worse than for the rest of the protocols. Accordingly, the results for AODV are not shown in order to show in more details the performance differences among the other protocols. From Fig. 7.8 we see that there is no appreciable difference in the performance of the routing protocols, NEST, SOAR and DSR for the *m* and *s* model. This is because the radio range is 250 m and the netmark moves over an area of (250mx250m) for the *m* model, which does not contribute to too many additional path changes. The performance of all the routing protocols suffers when the netmark becomes highly mobile. From Fig. 7.8, we find that the INTNET model produces more stress on the routing protocols than the RELIEF model does, with DSR being affected the most. Though the traffic pattern is different in both cases, the total number of data packets sent throughout the simulation is the same and the network is not overloaded.
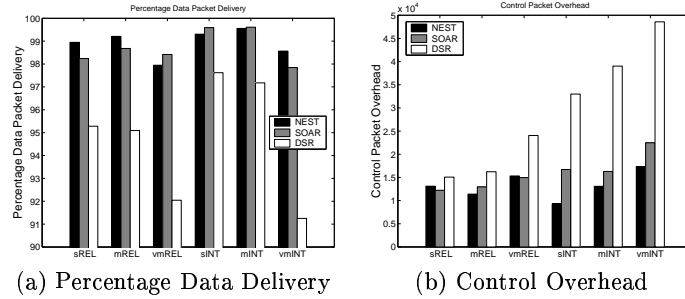

(a) Percentage Data Delivery   (b) Control Overhead

Figure 7.8.   Performance in a 31node Network with varying mobility models for netmark and two different traffic models

Table 7.3.   End to End Delay Distribution of the Voice Traffic For Mobile Relief Model (mREL)

Percentile	NEST (s)	SOAR (s)	DSR (s)
90	0.0091	0.0110	0.0087
95	0.1136	0.1503	0.0734
97	0.2682	0.3301	0.2226
range (max−min)	19.89	13.7335	19.329

From Fig. 7.8(a), we see that SOAR and NEST deliver on an average the same number of data packets in both traffic models, which we have also seen in our results of Sec. 7.5.3 for the low-load scenarios. DSR's percentage data delivery is 4%-7% smaller than the data delivery

Table 7.4. End to End Delay Distribution of the Voice Traffic For Mobile Relief Model (mREL)

Percentile	NEST (s)	SOAR (s)	DSR (s)
90	0.0101	0.0151	0.0121
95	0.1067	0.1768	0.0917
97	0.2150	0.3799	0.2087
range (max−min)	2.5041	14.4443	49.083

Table 7.5. End to End Delay Distribution of the Voice Traffic For Very Mobile Relief Model (vmREL)

Percentile	NEST (s)	SOAR (s)	DSR (s)
90	0.0611	0.0834	0.0986
95	0.3222	0.3930	0.3695
97	0.6832	0.8593	0.6371
range (max−min)	17.28	23.4054	13.119

achieved by SOAR and NEST, with the performance becoming worse with higher mobility of the netmarks. This is because of packet losses due to unavailability of routes to forward data packets at intermediate routers, which implies that DSR suffers due to stale path information.

DSR's control overhead is comparable to that of SOAR or NEST for the $sREL$ or $mREL$ models (Fig. 7.8(b)). DSR sends significantly more control packets for the INTNET model, where DSR utilizes redundancy in routing information less efficiently than SOAR or NEST.

SOAR and NEST have similar control overhead for the RELIEF model, though in the INTNET model NEST outperforms SOAR and DSR. This is because the RELIEF model has fewer flows (around six) towards the netmark compared to the INTNET model, in which theoretically any node can communicate any time with the netmark. When the number of flows is smaller, fewer links are used for active data delivery. Because detections of link failures are triggered only by the failure of transmission of data packets, with fewer flows more links remain stale in the topology table. Therefore, if the number of flows between common nodes is the same as the number of flows between netmarks and common nodes, the Hello mechanism does not improve the condition because SOAR and NEST require almost the same number of updates to purge wrong routing information. This indicates that the node-centric approach to proactive route maintenance can improve the performance of the network, however the degree of performance improvement de-

pends on the amount of communication between common nodes and the netmark.

We also observe that netmark mobility does not impact the performance of NEST more than the performance of purely on-demand approaches, which could have been an argument for using an on-demand approach rather than a hybrid approach when netmarks are very mobile.

Because voice traffic is delay sensitive, we analyzed the delay performance for each of the routing protocols (Tables 7.3, 7.4, 7.5) for the RELIEF traffic model, where voice traffic is used. The results presented are for a randomly chosen run, so as not to average out the high frequency components of individual runs. This is important for voice traffic performance, because worst-case performance results are required for quality assurance. The following conclusions can be drawn from the data available from Tables 7.3, 7.4 and 7.5:

- Range (the difference between minimum and maximum delay) is significantly high under all cases. This is because the network becomes partitioned, and the node discovery mechanism is not very fast in on-demand routing protocols and can become really slow as the timeouts for resending queries increases non-linearly. The range for DSR for Mobile Relief Model (Tables 7.3, 7.4 and 7.5) is as high as 49 secs. Though NEST maintains proactive routes with the netmark, the range for NEST is also high because it uses on-demand routes to common nodes.

- As expected, there is an increase in delay when the netmark is more mobile, because nodes have more stale routes.

- NEST has better delay performance than SOAR in terms of percentile values because NEST has fewer queries and the paths tend to be more up to date, thereby spending less time queueing packets either at the routing layer or the link layer. This indicates that the hybrid routing approach helps to reduce the end-to-end delay of data packets.

7.5.4   Experimental Scenario 3

The third scenario is designed to address the impact of multiple netmarks on the performance of hybrid routing solutions. This scenario has two netmarks along with 30 mobile nodes. The netmarks are placed at coordinates (250, 250) and (750, 250), and are static throughout the simulation. The traffic pattern is according to the INTNET model. Packets from the Internet can enter the ad hoc network through any of the two netmarks. Because the netmarks advertise routes to the Internet for the
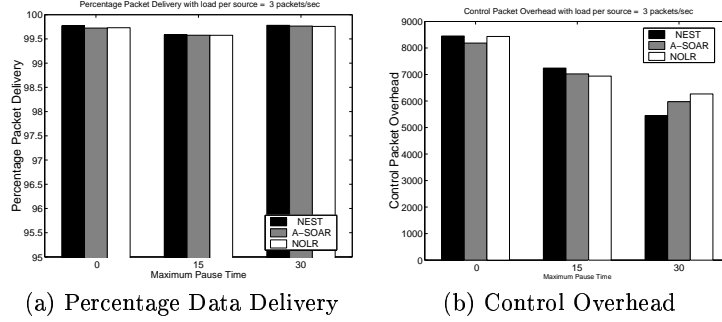
(a) Percentage Data Delivery   (b) Control Overhead

Figure 7.9.   Performance of NEST, SOAR and NOLR for a network with 30 nodes and 2 netmarks

entire ad hoc network and not individual host routes, it can be safely assumed for the experiments that the incoming flows are randomly distributed between the two netmarks. Packets for the Internet are always forwarded to the nearest netmark. After a node decides which netmark to use, it encapsulates the IP packet with an IP header with the destination being the netmark's address. When routes to any netmark are not available, anycast queries (as discussed in Sec 7.4) are sent and routes are established based on the *anycast* replies. We compare the performance of NEST with SOAR (denoted as anycast-enabled SOAR or A-SOAR in Fig. 7.9) and NOLR. Like A-SOAR, if needed, NOLR and NEST also use *anycast* queries for netmarks.

From Fig. 7.9, we see that SOAR and NOLR both perform as good as NEST in terms of data delivery and control overhead under all three mobility scenarios. This is in contrast to the results presented in Sec. 7.5.2 and Sec. 7.5.3, where both SOAR and NOLR produce higher control overhead than NEST. This performance improvement in SOAR can be attributed to the following three reasons: (a) If a route to a given netmark is not available, packets can still be sent to other netmarks, which helps in reducing the number of queries; (b) anycast route replies help to prevent flooding of queries and faster route discovery; and (c) reducing the number of *reply-query* packets helps to prevent old link-state information to be injected into the network, which helps to reduce the number of updates. However, the proposed node-centric hybrid approach with proactive routes for netmarks is still attractive for this small network if the static or the hybrid method (Sec. 7.4) is used for forwarding outgoing packets, because in such situations every node might need a path to every netmark.

7.6.   Conclusions

We have presented node-centric approaches to hybrid routing for ad hoc networks that distinguish between normal nodes and special nodes called netmarks, which host popular network services or function as points of attachment to the Internet. With node-centric hybrid routing, netmarks force other common nodes to maintain routing information for them by either advertising their routing information as in table-driven routing protocols, or by requiring nodes to maintain routing entries towards them for extended periods of time. This reduces the network-wide flooding and the corresponding delay for route set up every time a session needs to be established between a normal node and a netmark. Routes between peer nodes are set up on-demand. We have evaluated the changes needed to incorporate node-centric hybrid routing in the basic mechanism of routing for some pure on-demand routing protocols, namely AODV, DSR and SOAR and compared the performance of AODV, DSR and SOAR with the hybrid approaches, NEST and NOLR (which have been adapted from SOAR) using ns2.

Our findings show that DSR's control overhead is higher than AODV's for highly mobile and highly non-uniform traffic patterns. This is in contrast to results of the previous studies [Das et al., 2000], [Broch et al., 1998], where performance evaluations have been done with uniform traffic pattern.

On the basis of ns2 simulations, we have found that, if a node in the ad hoc network acts as a source or relay of data packets for significant portion of its lifetime, the benefit of extending caching information in a purely on-demand routing protocol is not noticeable. However, maintaining proactive routes as in NEST offers better performance than any on-demand routing protocol, both in terms of data delivery and control packet overhead when the traffic flow is mostly from common nodes towards the netmark. We have also found that the performance of NEST is not affected by the mobility of netmarks. In a moderately-sized network served by multiple netmarks, the performance of on-demand routing protocols can be significantly improved by maintaining routes to any of the netmarks and then sending anycast queries asking for a route to the nearest netmark.

References

[Broch et al., 1998] Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y. C., and Jetcheva, J. (1998). A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. ACM Mobicom.

[Das et al., 2000] Das, S. R., Perkins, C. E., and Royer, E. M. (2000). Performance Comparison of Two On-Demand Routing Protocols for Ad-Hoc Networks. IEEE Infocom.

[Haas and Pearlman, 1999] Haas,   Z.   and   Pearlman,   M.   R.   (June 1999).   The   Zone   Routing   Protocol   (ZRP)   for   Ad   Hoc   Networks. http://www.ee.cornell.edu/ haas/Publications/draft-ietf-manet-zone-zrp-02.txt.

[ISI-NS, 2000] ISI-NS   (25   May,   2000).   The   network   simulator - ns-2.1b6, http://www.isi.edu/nsnam/ns/.

[Johnson and Maltz, 1994] Johnson, D. B. and Maltz, D. A. (1994). Dynamic Source Routing in Ad-Hoc Wireless Networks. Mobile Computing.

[Marina, 2000] Marina, M. (last updated on 12/07/2000). Aodv code for cmu wireless and mobility extensions to ns-2. http://www.ececs.uc.edu/ mmarina/aodv/.

[Perkins et al., 2002] Perkins, C. E., Royer, E. M., and Das, S. R. (March, 2002). Ad Hoc On-Demand Distance Vector (AODV) Routing. Mobile Ad Hoc Networking Working Group, draft-ietf-manet-aodv-10.txt.

[Roy and Garcia-Luna-Aceves, 2001] Roy, S. and Garcia-Luna-Aceves, J. J. (2001). Using Minimal Source Trees for On-Demand Routing in Ad Hoc Networks. IEEE Infocom.

[Royer et al., 2001] Royer, E. M., Sun, Y., and Perkins, C. (2001). Global Connectivity for IPv4 Mobile Ad hoc Networks. draft-ietf-manet-globalv4-00.txt.

[Tsuchiya, 1988] Tsuchiya, P. F. (1988). The Landmark Hierarchy: a New Hierarchy for Routing in very Large Networks. ACM Sigcomm.